Introduction to Programming
Week 6 tutor’s kit

1. The situation

Theses for this week:

• testing, more code cliches
• group skills, running a meeting, code quality, code review, code as communication with people and machines

This is the first real week on Problem 2 and students should be working very hard on the prototypes and acceptance tests. It is important to make it clear that this is a chance for them to get a clear indication of what they plan to do and to show that in terms of the prototypes and acceptance tests. They should look at the marking scheme and they should not panic. The first stage of the project is, above all else, a way for them to get solidly started with us checking that they taking a safe road. The semester plan shows that this week they should be showing these to tutor’s for test marking.

Record the marks for the tute and pass on summaries of how students went to your section leader. You should have a fairly complete set of marks for Problem 1 by now. Use inmarks to create the electronic marks set. Netfile it to your section leader.

2. Tutorial script

<table>
<thead>
<tr>
<th>Activity 0</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline class:</td>
<td>minutes</td>
<td>minutes</td>
</tr>
<tr>
<td>Main activities are:</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>• quiz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• group skills and planning activity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• work on prototypes - planning and discussing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You will need to sell the merits of the first two of these.

<table>
<thead>
<tr>
<th>Activity 1</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1. Quiz:</td>
<td>minutes</td>
<td>minutes</td>
</tr>
<tr>
<td>Write questions on the board from the quiz at the end of this kit.</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Adapt it to your class.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Take care to link it to the task of making structural prototypes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Also learning and consolidating Blue concepts of scope, interface, routines, parameters, instance variables, quality of comments, code clarity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1.2. Mark the quiz

This follows the pattern started last week.

You ask students to exchange answers and mark each others

This means they get their heads around another person’s code

Weak students may see a good solution.

I would then collect these and when there is a period when students are working on another activity, I would record the results and just look at what the answers are like for each student. The idea is that these quizzes give you an idea of how each student is doing.

<table>
<thead>
<tr>
<th>Activity 2</th>
<th>time for it</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>minutes</td>
<td>minutes</td>
</tr>
</tbody>
</table>
2. **Group skills and planning activity**  
Ask students to read about roles in groups (Kingston Ch 11)  
Ask each group to allocate these roles.  
Ask for a written version of this so you can keep it.

**Post:**  
You will have spent time with each group and for each, reviewed their plans and will now be confident that each has a safe plan of action

- some is associated with each role on p124-5 ***  
- you know who is librarian  
- 1-2 people will be assessment experts - reading assessment carefully **  
- 1-2 people allocated as domain experts - learning about it **  
- 1-3 students working on functional prototype **  
- 1-3 students working on structural prototype **  
- 1-2 students working on acceptance tests **  
- you are sure that no very weak student has a critical task to do alone  

** means these subgroups should plan to meet and work together  
also each group needs:  
Unix expert(s)  
assessment expert(s)

<table>
<thead>
<tr>
<th>Activity 3</th>
<th>time for it minutes</th>
<th>total time minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work on prototypes</td>
<td>25-45</td>
<td>110</td>
</tr>
</tbody>
</table>

This is what they will be wanting to do!!

---

### 3. **3-hr Workshop**

<table>
<thead>
<tr>
<th>Activity 1</th>
<th>time for it minutes</th>
<th>total time minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class overview</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**Class overview**  
Explain how you will spend your 1 hour of presence.  
Your main role is to give feedback on prototypes and plans.  
You should ask students what they plan to do in the 3-hrs.

Their main task is to get through aspects that need the whole group present  
They should focus on  
• agreeing on how to split work  
• reviewing what others have done  
• once agreed on work split, working on allocated tasks

<table>
<thead>
<tr>
<th>Activity 2</th>
<th>time for it minutes</th>
<th>total time minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students work</td>
<td>rest</td>
<td>rest</td>
</tr>
</tbody>
</table>

**Students work**  
You review plans and look at prototypes so far.  
You give feedback on both.  
**Post:** You have a copy of each group’s planning sheet for the coming week -  
You will bring this back next week to discuss progress.  
You will be reasonably happy with the plan  
You are reasonably happy with the work split  
Everyone should be writing code - for one prototype or the other  
(They need to practice to learn to program)
Quiz for week 6

These questions relate to the code in the COMP1001/1901 Resource Book. We want to focus on two aspects of the class Floor

They are intended to help students consolidate what they have been learning about class interfaces. When going through answers, please emphasise the relevance of this stuff to the design of the structural prototype.

There is also one part that touches on scope - make sure you consolidate their growing understanding of visibility of variables - talk about the instance variables and their invisibility outside the class.

Question 1.

What would appear at the interface view of the class? (student can highlight the relevant parts or otherwise mark them)

(Answer is here for tutor's reassurance)

class interface Floor is
=================================================================================================
== Exists to hold lists of people, and to interact with elevators.
=================================================================================================
creation (building : Building, level : Integer) == Creation routine comment here
routines
    toString -> (s : String) == printout
    addPerson (p : Person) == add to our list(s)
    getPasenger (direction : Direction) -> (result : Person) == get the next passenger in line who’s going in 'direction'
    == Returns 'nil' if there’s noone.
    peopleWaiting -> (result : Boolean) == anybody waiting here?
end class

Question 2.

What are the instance variables?

Answer:

building: Building
level : Integer -- ground floor is level 1
goingUp: LQueue<Person>
goingDown: LQueue<Person>
end class
Question 3.
How would another class create a 5th floor for a Madsen Building?

Question 4.
Write the code that another class might use to print the message "Stop at this floor" if there are people waiting at that floor and "Skip this floor" otherwise.

Question 5.
Suppose another class had code that had to print how many people are waiting at each floor (ie how many are in goingUp or goingDown. What would they need to do to achieve this.

Answer: Probably best to write a new routine in this class. Kludge would be use getPassenger repeatedly counting how many you can get before the list is empty, and do this for both directions.

These instance variables are invisible outside the class.

Question 4.
Critique each element of the interface - those present and absent.

Answer:
• class name is pretty good for a class that represents a floor
• class comment is fairly clear
• comment on toString is not helpful
• addPerson - would be better as add a person ‘p’ to the appropriate lift queue - either up or down
• getPassenger seems pretty clear to me - parameters and return result have good names and the Class names Direction and Person are helpful
• peopleWaiting also perhaps a better name for the return variable is something like ‘people_there’
• no pre or post conditions !!!! That may be ok for a prototype with a single author. Sell the merits of them for multi-author projects.

Judy Kay
1999 PBL Co-ordinator for Comp1001/1901
April 1999

PS. Please send feedback to your section leader.